Universidad Rafael Landivar
Facultad de Ingeniería
Ingeniería en Informática y Sistemas
Estructura de datos II - Sección 1
Catedrático: Ing. Fredy Alexander Buscamante

PROYECTO DE APLICACIÓN:

SISTEMA DE GESTIÓN DE ARCHIVOS SEGUROS Y EFICIENTES

Alland André Gonzáles Franco: 1233022
Joaquín Raymundo Choc Salvador: 1280423
Daniel Alberto Díaz Pérez: 1040023
Hector José Flores Pineda: 1199923

Guatemala, 4 de noviembre del 2024

ÍNDICE

DESCRIPCIÓN DE LA APLICACIÓN

El programa desarrollado tiene como objetivo facilitar la gestión de grandes volúmenes de manera eficiente, mediante la compresión y encriptación de datos con el fin de optimizar el espacio de almacenamiento y mejorar la seguridad de la información de los clientes.

Esta aplicación cuenta con las siguientes funcionalidades:

Almacenamiento Seguro y Optimizado

El programa permite al usuario almacenar archivos individuales o carpetas completas, ofreciendo las siguientes opciones:

- **Encriptación**: Protege los datos encriptándolos mediante una contraseña.
- **Compresión**: Optimiza el espacio de almacenamiento usando algoritmos de compresión (LZW y Huffman).
- **Ambos**: Aplica las operaciones de encriptación y compresión al mismo archivo o carpeta, logrando una mayor seguridad y optimización.

Selección de Archivos o Carpetas:

- **Archivo individual**: El programa genera una versión encriptada y/o comprimida del archivo con una extensión específica según las operaciones aplicadas.
- **Carpeta completa**: El programa crea una nueva carpeta en la que cada archivo tendrá una extensión ajustada según las operaciones realizadas (compresión y/o encriptación), permitiendo identificar las modificaciones aplicadas a cada archivo.

Compresión de Archivos

Permite que el usuario pueda seleccionar el tipo de algoritmo de compresión, donde se le solicitará entre los dos algoritmos de comprensión implementados (LZW y Huffman), así mismo, se brinda la opción de seleccionar ambos algoritmos, con el fín de lograr una compresión eficiente sin pérdida de datos.

Encriptación de Archivos

Los archivos son encriptados utilizando una contraseña definida por el usuario, dicha contraseña es necesaria, porque será utilizada para la desencriptación de los archivos en cuestión y así acceder al contenido del archivo o carpeta, garantizando la seguridad y confidencialidad de los datos.

Recuperación de Archivos

El programa permite recuperar archivos previamente comprimidos y encriptados mediante el análisis de su extensión, identificando así las operaciones necesarias para restaurar la información. Además, el usuario podrá especificar la ubicación de destino para el archivo o carpeta recuperada, brindando mayor control y organización en la gestión de archivos.

Generación de Archivo LOG

El programa genera un registro detallado después de cada operación realizada, en la misma ubicación donde se guardaron los archivos, que incluye la siguiente información:

- Nombre original del archivo
- Tiempo de ejecución de cada operación (compresión, encriptación, etc.)
- Tasa de compresión lograda (cuando se aplique compresión)
- Nombre del archivo generado

Este registro facilita el seguimiento y control de los cambios realizados en cada archivo, para que se pueda observar la efectividad y el rendimiento de las operaciones de comprensión y/o encriptación realizadas en cada archivo.

EXPLICACIÓN DE LA CODIFICACIÓN Y GENERACIÓN DE ARCHIVO COMPRIMIDO

Al seleccionar la opción de compresión, se solicitará al usuario que ingrese el archivo o carpeta a comprimir y la ruta donde se exportará el archivo resultante. El usuario puede elegir entre dos modos de compresión: LZW o LZW + Huffman. Dependiendo de la elección del usuario, se ejecutarán diferentes funciones:

LZW:

Se invoca el método estático CompressWithLZW de la clase CompressionDecompression,, el cual recibe como parámetro un array de bytes llamado data. Este método comienza inicializando un diccionario que asocia cada carácter ASCII con su valor correspondiente.

Para comprimir, se sigue el siguiente proceso:

1. Se utiliza una variable llamada *current* para almacenar la secuencia de caracteres *actual*.

2. Mientras se procesa cada byte en *data*, se construye una nueva secuencia *next* que combina *current* con el nuevo carácter (byte).

3. Si *next* ya existe en el diccionario, la secuencia se expande y *current* se actualiza para que sea igual a *next*.

4. Si *next* no está en el diccionario, el código de *current* se agrega a la lista *compressedData*, que representa la secuencia comprimida hasta ese momento.

5. Si el tamaño del diccionario no ha alcanzado su límite, *next* se agrega al diccionario con el siguiente código disponible, comenzando desde el código 256.

Finalmente, el método convierte *compressedData* en un arreglo de bytes, donde cada código es representado como un entero corto (short) de 16 bits, y retorna este arreglo comprimido.

Método LZW + Huffman:

Al seleccionar esta opción, primero se invoca el método CompressWithLZW de la misma forma que en el método LZW, obteniendo un array de bytes comprimidos. Este array se pasa luego al método estático CompressWithHuffman, también de la clase CompressionDecompression.

El proceso de compresión con Huffman se lleva a cabo de la siguiente manera:

1. Se construye un mapa de frecuencias de los caracteres en el array de bytes.

2. A partir de este mapa, se genera un árbol binario de Huffman. Las rutas en el árbol hacia cada carácter determinan la codificación en 1's y 0's: a medida que se navega en el árbol, se registra un 1 o 0 según la dirección tomada (hacia la izquierda o la derecha) para llegar a cada carácter.

3. Este árbol permite representar cada carácter con una secuencia de bits optimizada según su frecuencia.

Exportación del Archivo Comprimido

Después de realizar el proceso de compresión, se utiliza la clase File y el método WriteAllBytes, el cual toma como parámetros los datos comprimidos y la ruta especificada para almacenar el archivo comprimido.

EXPLICACIÓN DE LA ENCRIPTACIÓN Y GENERACIÓN DE ARCHIVO ENCRIPTADO

Al elegir la opción para encriptar, se le solicitará al usuario el archivo .txt o carpeta al que se le aplicará la operación y la ruta para exportar el archivo generado, así mismo se le solicitará una contraseña que será la llave para desencriptar el archivo o carpeta que se encriptó anteriormente.

El proceso que realiza la opción encriptar se toma de la siguiente manera:

- **EncryptData(byte[] data, string password)**

  Método principal donde coordina el proceso de encriptación, mediante una estructura que aplica varias transformaciones para ofuscar los datos

Así mismo se detalla los pasos técnicos:

- **Header**

  - Se Convierte en bytes utilizando el método “Encoding.UTF8.GetBytes”
  - Seguidamente se crea un arreglo “dataWithHeader”, que contiene el header como los datos originales del archivo .txt o carpeta (data).
  - Así mismo se usará en la desencriptación para la verificación de integración del archivo y asegurar que la contraseña proporcionada es correcta.

Este proceso consta de tres operaciones secuenciales: XOR, transposición y sustitución de bytes.

- **XOREncrypt (XOREncrypt(byte[] data, string password))**

  - **Entrada:** se utiliza el arreglo “dataWithHeader” para la combinación del header y los datos originales (data).
  - **Generación de contraseña:** La contraseña se convierte en un arreglo de bytes “passwordBytes” para realizar la operación XOR.

  Proceso XOR:

  - Cada byte que contenga “dataWithHeader” se utilizara la operación XOR con el byte correspondiente de “passwordBytes”.
  - Si la longitud “passwordBytes” tiene una menor longitud que “dataWithHeader”, el índice comienza un ciclo usando el operador “(i % passwordBytes.Length)”, esto creando un patrón repetitivo.
  - **Resultado:** Se generá un arreglo de bytes “encryptedData” que depende tanto del contenido de “dataWithHeader” como de la contraseña.

- **Transposición de Datos (Transpose(byte[] data))**

    - **Entrada:** Se utiliza el arreglo de bytes “encryptedData” para la función de transposición.
    - **Inversión de Orden:** Se utiliza “Array.Reverse” para invertir el orden de los bytes del arreglo “encryptedData”.
    - **Resultado:** El arreglo transpuesto introduce un segundo nivel de complejidad, debido a que al momento de desencriptar el archivo o carpeta el orden debe ser restaurado.

- **Sustitución de Bytes (SubstituteBytes(byte[] data))**

    - **Entrada:** Se utiliza el arreglo “encryptedData” después de pasar por el proceso de transposición:
    - **Generación de S-box (GenerateSBox):** Este método crea una tabla de sustitución de 256 bytes (0 a 255), en la cual cada índice se reemplaza por un nuevo valor calculado como (i * 31) % 256. Esto garantiza que cada byte tenga un valor único entre 0 y 255, proporcionando una sustitución eficaz.
    - **Aplicación de S-box:** Cada byte generado por el arreglo “encryptedData” es reemplazado con el valor en la posición correspondiente de “S-box (substitutionBox[data[i]])”.
    - **Resultado:** El arreglo de bytes encriptados de “encryptedData” contiene los tres métodos anteriormente mencionado

CONCLUSIONES

El programa implementa una solución integral y altamente eficiente para la compresión de archivos, optimizando de manera significativa el uso de espacio de almacenamiento en los servidores de la empresa, minimizando los costos asociados con el almacenamiento y mejorando la capacidad de la infraestructura para gestionar volúmenes crecientes de datos sin comprometer el rendimiento.

Asimismo, la incorporación de encriptación de datos en el proceso añade una capa esencial de seguridad, garantizando la confidencialidad de la información sensible de los clientes, este mecanismo de protección es fundamental para evitar accesos no autorizados y proteger los datos ante potenciales riesgos de seguridad, la encriptación fortalece el cumplimiento de normativas de privacidad y seguridad de datos, lo que contribuye a mejorar la confianza de los clientes y a reducir el impacto de posibles incidentes de seguridad, alineándose con las mejores prácticas de la industria.

Además, el programa incluye una funcionalidad de recuperación automática basada en la detección de la extensión de los archivos, lo que permite a los usuarios revertir las operaciones de compresión y encriptación de forma intuitiva y sin necesidad de procedimientos complejos, esta capacidad no solo agiliza la gestión de archivos y minimiza los errores durante la restauración, sino que también ofrece a los usuarios una experiencia de uso simplificada, adaptándose fácilmente a los flujos de trabajo.

En conjunto, este sistema avanzado de compresión, encriptación y recuperación automática se traduce en un manejo de archivos más seguro, eficiente y accesible, ayudando a la empresa a optimizar sus recursos, proteger la información de sus clientes y simplificar la gestión de datos en el entorno digital.